OPTIMIZING PERFORMANCE IN INTELLIGENT ARCHIVES

23 January, 2003

H. STEPHEN MORSE
DAVID ISAAC

Business Performance Systems
7364 Clifton Road
Clifton, VA 20124

CHRISTOPHER LYNNES

Code 902
NASA/GSFC
Greenbelt, MD 20771

Abstract

Space-based remote sensing systems continue to provide valuable long- and short-term data sources for Earth science research and related applications. A critical component in such a system is the *data archive* which stores the data collected by the remote sensors (perhaps following a value-added processing step), and then makes this data available to scientists and other users. Looking ten to twenty years into the future, it is likely that the current practice of manually optimizing the operations of archives will not suffice as systems become increasingly complex and data users become increasingly diverse. This is particularly true for NASA's Earth Science Enterprise as it embarks on an initiative to increase the use of Earth science data for a variety of non-research applications, many of which have large user communities and stringent near-real-time requirements. By adding intelligence to the data archive, it may be possible to substantially improve the level of service provided by the archive while reducing costs by better anticipating user data requests and making more effective use of available resources. We expect a variety of benefits to users, the most important of which is reduced latency of product delivery for time-sensitive applications.

This paper is one of a series of studies focusing on various aspects of an *intelligent archive* (IA). This work is sponsored by NASA's Intelligent Systems Project within the Computing, Information and Communications Technology (CICT) Program. A conceptual study of intelligent archives of the future is underway[1], and the work reported here is intended to contribute to that effort. Here, we will focus on issues concerned with optimizing performance.

This paper begins (§1) by clearly defining the problems that will face data archives for remote sensing systems. On the one hand, sensors are increasing in capability, requiring corresponding increases in bandwidth, storage capacity, tasking strategies, and algorithmic and processing sophistication and complexity. On the other hand, the user community and associated applications consuming this data – and their requirements – will continue to expand. Reduced latency (perhaps as low as tens of minutes) is only the most visible of a host of requirements for improved efficiency, functionality, and product quality. We then discuss, at a functional level, how intelligence in the archive might address these problems, and illustrate this with an example scenario based on fires in the Western United States.

Next (§2), we examine the applicability of a variety of candidate technologies to these requirements. In addition to approaches involving machine learning, optimization, and scheduling, we discuss a novel approach coming from researchers in control theory. We show how *Model Predictive Control* (MPC) might serve as a conceptual unifier in developing a systems-level integrated solution framework. We then discuss (§3) seven significant technical challenges facing an intelligent archive supporting remote sensing systems. Finally, we briefly summarize our conclusions and recommendations (§4). An appendix provides additional background on MPC for the interested reader.

Contents

1. PROBLEM STATEMENT

The following four subsections respectively address the role of the archive in the overall system architecture; the major drivers that motivate the inclusion of intelligence in the data archive; a brief characterization of the term *intelligence* as applied to an archive; and an example scenario that highlights the functional capabilities of an intelligent archive and provides a glimpse into its associated concept of operations.

THE ROLE OF THE INTELLIGENT ARCHIVE

To provide context for the ensuing discussion, we begin with a brief overview of typical remote sensing programs supported by a future intelligent archive, and the role the archive plays in them. The intelligent archive is likely to be a major piece of an overall end-to-end knowledge-building system, which would no longer be confined to a single location. Rather it is a distributed entity comprising components fulfilling a variety of functions (principally data understanding, management and persistence) at potentially dispersed locations, with efficient interfaces to other pieces of the end-to-end system such as intelligent sensors and processing systems. The distributed nature of this system, together with the need for efficiency in intra-system interactions, requires robust resource management.

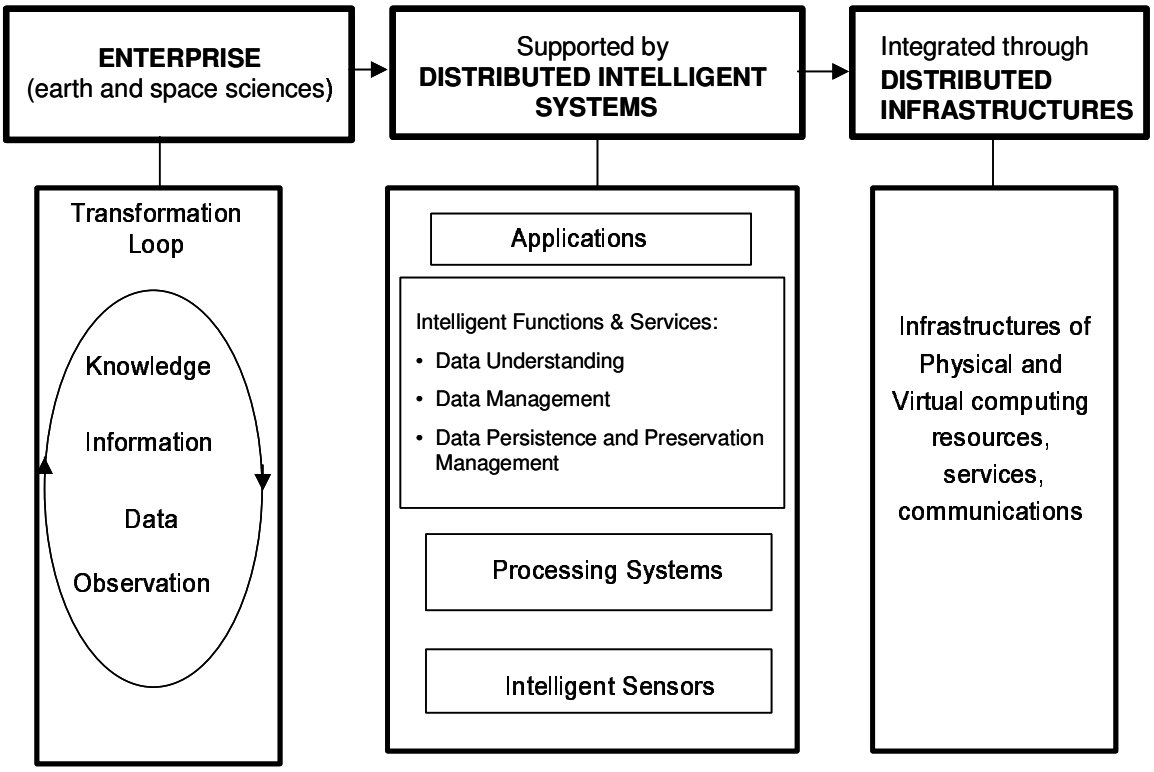

Figure 1. Context diagram for intelligent archive.

The resource management domains within this system are depicted along with conceptual data flows in Fig. 2. First (1), there are the *physical and geophysical processes* that are the target, or object, of collection. While these do not constitute a resource management domain themselves, they will be shown to have a potentially powerful effect on resource management decisions elsewhere in the end-to-end system. Second (2), there are the *data collectors* – that is, the collection vehicles and sensors that remotely and digitally sample the signals and down-link the data to ground-based receiving stations. Third (3), there are the *archival facilities*, which are the principal focus of this paper. Fourth (4), there are the *data producers* (including both production processing and science processing facilities) that process the data into estimates of geophysical parameters and higher-

level data products. And fifth (5), there are the *users* and their applications that request and use the data stored in and provided by the archive. These resource management domains are functionally but not necessarily physically separate: in many cases some production processing may be done at the archival facility to save cost; likewise, the users may consist of other data collectors or data producers. Conversely, a given resource management domain may be widely distributed.

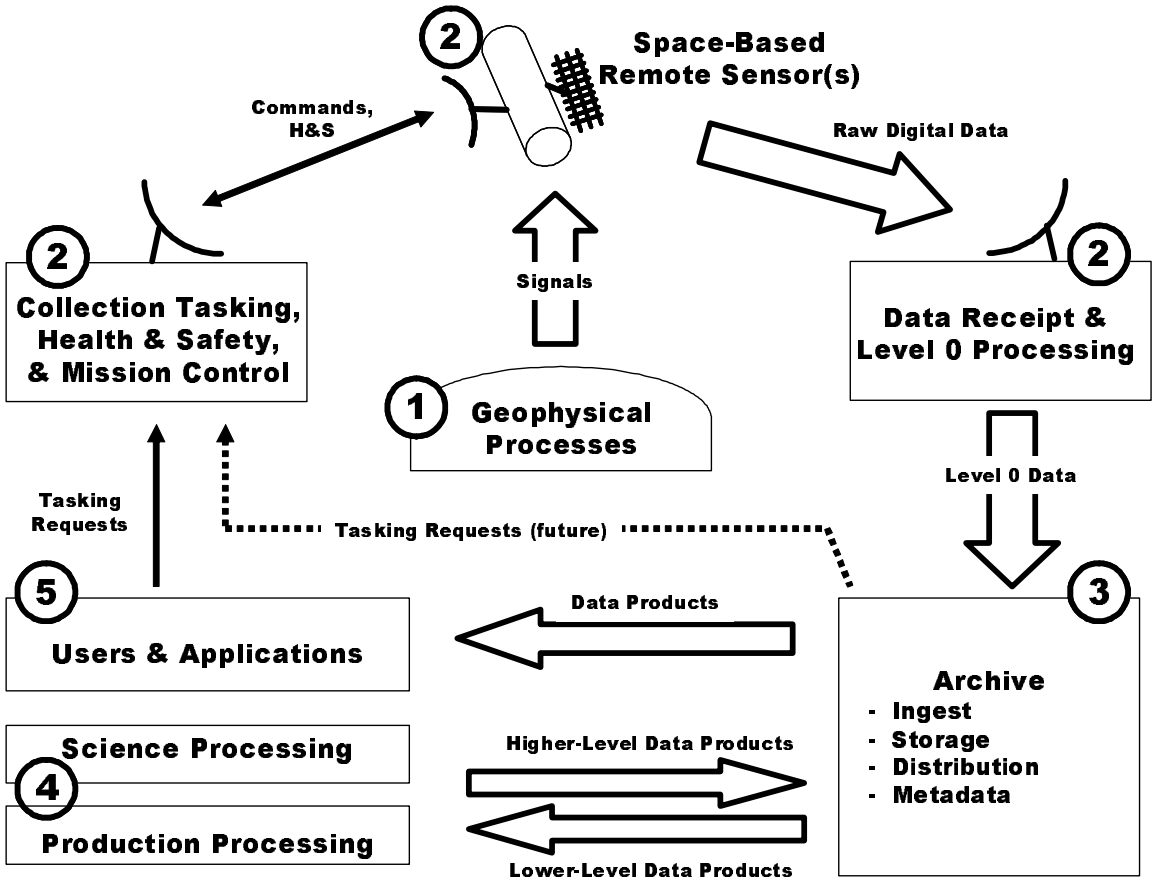

Figure 2. Resource management domains in the end-to-end system, with conceptual data flows.

Management of collection tasking for a sensor on an orbiting vehicle is typically the responsibility of an instrument team. Many/most sensors are currently down-staring and not pointable; the only available commands are on- and off-times; and when on, the sensor collects everything in its field of view. A few other sensors (and their proportion may increase over time) are pointable, so that more detailed collection plans are required on a pass-by-pass basis. In present-day systems, the data generated by the sensor is stored in on-board buffers for periodic (once-per-rev) downlink (although real-time X-band broadcasts may also be supported). In the typical case, the data is sent to a facility for Level-0 processing, and then shipped directly to the archive. This latency (assuming deferred downlink together with Level-0 processing) may be as much as 90 minutes. Future systems are likely to have more processing capability and even some archive capability, re-emphasizing the idea that these resource management domains are functionally determined. However, space-based resources are inherently more limited due to weight, power and radiation; thus, resource management domains can be strongly influenced by physical location.

Once the data have arrived at an archive facility, some production may follow. Depending on requirements, this may include routine processing into higher level products (Level 1-4) and creation of metadata. Data

may be shipped directly to customers (if low latency is important), or a more leisurely production and delivery schedule may be applied. In present-day systems, many customers have standing orders, and latencies on the order of two weeks are often acceptable. Further, delivery of the product on tape cassette is frequently adequate, thereby reducing demands on network bandwidth. However, future systems are likely to require lower latency as more users exploit the data as a decision support tool.

In addition to standing requests, users make *ad hoc* requests for archived data and request specialized processing (or re-processing) of that data. These requests are often event-driven. For example, the eruption of a volcano may trigger heavy demand for data from many sensors at that location over a long period of time – up to several years. Other request patterns exhibit temporal periodicity related to the academic year and research intensity. These *ad hoc* requests add to the total load on the archive, and can become a problem if a large number of requests cluster over a short period of time.

Another type of request is focused on very recent collections, and low latency is an important requirement. Perhaps 10% of current resources are devoted to these types of near-real-time applications, and this percentage is expected to increase. For these requests, which may be standing or *ad hoc* requests, the archive must expedite product delivery independent of its archival functions. In this capacity, it acts as a value-added store-and-forward buffer, generating specialized products which are rapidly relayed to the end customer, typically by WAN. Often, in the current environment, the production sequence for these products can be predefined and triggered at ingest. In the future, it may be necessary to select the appropriate processing flow dynamically, perhaps adjusting it in mid-stream based on content-processing that searches for triggering events in the data itself.

The significant resources within the archive include ingest communications equipment (and associated bandwidth); the large tape archive (including multiple drives and tape management robotics); data processing equipment (high performance processors for production); a high-performance disk cache for staging data between tape, production, and ingest/export; and interfaces to WANs and the Internet for product delivery as well as for connectivity to other archives and processing facilities. This last item is important, since some concepts of operations for future architectures include the ability to distribute processing and storage loads over multiple sites and facilities. In this scenario, one option available to an archive (for example, to satisfy a particular request) might be to ship local data to a remote facility where it is fused with data from other sources, specialized processing performed, and data delivered. Alternatively, if production requires access to data held at another facility, this data may first be staged or shipped to the archive prior to processing. These and other production scenarios are actively under consideration for future implementation. One area of possible optimization is for the archive itself dynamically to select among these options on a case-by-case basis depending on current or predicted system state and loading.

The intelligent archive itself does not include processing as an integral function, though many requested products require further (on-demand) processing. However, this processing may best be considered as a service which the archive *requests or supports*, rather than a service which it *directly provides*. In this view, the archive serves the role of intelligent broker and coordinator, orchestrating the activities of service providers to achieve a complex product (the result, for example, of data from multiple sources that has been subject to a complex and possibly conditionally branching processing flow). In the discussion which follows, we clearly emphasize the coordination role of the archive as an *efficient requestor* of data and services. However, the intelligence to optimize this coordination and scheduling role must include state and dynamical models of the processing services being requested as well as some ability to control and task them. In actual practice, if the processing resources are collocated with the archive, there is a tendency to blur the distinction between requesting a service, on the one hand, and directly tasking an owned resource, on the other. This issue, which is partly architectural and partly administrative, lies beyond the scope of this paper. Our focus will be on improved functionality, adaptability, and performance (latency, bandwidth, priority, value) achievable by adding various sorts of intelligence to what is, now, a fairly rigid and inflexible archival capability.

DRIVERS AND MOTIVATION

With the prior section as background, we can now discuss in greater detail the drivers that lie behind the interest in adding intelligence to the data archive. We shall briefly discuss nine such factors.

real-time or near-real-time applications: The percentage of users demanding low-latency product delivery is increasing. These include, for example, emergency response organizations that rapidly address or contain crisis situations. These customers may need tailored data products – that is, products that are the result of specially formulated scientific modeling algorithms and that must meet preexisting interface requirements – to sense the occurrence, location, extent, and severity of the event. Because of its position as the first recipient of raw data, and because of the proximity to high performance computing equipment, the archive is an attractive candidate for providing such a service. The problem, however, is that the need for this service is dynamic and conditional. This means that the archive must be flexible and adaptive, with the ability to adjust or coordinate processing flows and resources dynamically in response to stochastic events. Further, this capability must be provided with minimal disruption to ongoing standard production cycles and product delivery. Examples of such applications include precision weather forecasting, precision agriculture, and natural disaster management (including forest fires and flooding).

production disruptions: Considerable effort in current production management is devoted to responding to breakage or errors in the data and processing flow. For example, if a collection of data sets is ordered from another source, the application may require the complete set to begin production: partial deliveries must be handled intelligently depending on whether items were dropped due to errors or because the requested data were not available, and depending on whether or not the data are available from another source. Similarly, internal data errors may trigger a processing halt or abort, perhaps after significant processing has occurred. While intelligence, *per se*, may not be adequate to address these error- and quality-related disturbances, intelligence should have models for the underlying rates, and take them into account in considering job scheduling and distributed job execution strategies.

tip-offs, cuing, and rapid retasking: Future concepts of operation demand near-real-time retasking in response to events or circumstances detected by specialized content-based processing in the archive (that is, the event detector *cues* the collector). Here, we imagine an ongoing surveillance function, perhaps itself initiated based on increased likelihood of occurrence of an event with scientific or societal importance. As soon as the event is detected (e.g., as the result of event-recognition algorithms executed in surveillance mode for geographic regions known to be at high-risk), it is useful to rapidly obtain additional high resolution sensor data at the event location. This cueing must be low-latency, and hence must by-pass ordinary organizational review and approval processes. As should be apparent, this retasking will, itself, result in additional loads at the archive as the newly tasked data arrives for specialized processing and low-latency product delivery. This capability results in a stochastically driven feedback loop, with the associated complexity and potential for pathologies.

virtual products (on-demand processing): An archival capability currently under consideration is *virtual products*, created through on-demand processing. In this strategy, rather than process all derived products at ingest in order to store the results for future retrieval, the archive will store only raw data, performing the higher-level processing once the product itself has been specifically requested by a user. In this case, the impact on the archive is more dynamic and unpredictable loading, at least when compared to the very rigid and predictable loading typical of current operations. This capability requires some degree of optimization and data staging by the archive. Rather than simply rely on a preset processing schedule, it must dynamically examine the current and predicted loading, stage data needed for the product, and optimally schedule a potentially dynamic set of resources among competing active requests.

content-based processing, metadata, and retrieval: The archive is ideally placed for characterization of ingested data, e.g. for the purposes of quality assessment or the production of metadata for use in content-based retrieval. The decision of whether and when to perform such processing, the feedback implications of doing so, and the problem of dynamically balancing the added processing load against other requirements is a complex optimization problem. In addition, the event-detection algorithms may benefit from machine learning approaches to pattern recognition. This reflects the desire to replace computationally expensive, model-based scientific codes by much faster non-model-based (machine-learning) algorithms derived from training sets for the purposes of data management and applications decision support (if not for scientific research purposes).

efficient processing of event-based loading spikes and bursts: The occurrence of certain types of events, such as volcanic eruptions or earthquakes, can trigger heavy loads at the archive as the result of requests for historical data near the event location. The loading resulting from these "spikes" has the potential to disrupt normal processing flows. By understanding the characteristics of these request patterns, and/or by predicting them (say, by anticipating the occurrence of the event itself), the loading profile can perhaps be somewhat leveled, with the result of less disruption and reduced latency of the delivered products. Similarly, requests for data can potentially be grouped for efficiency. Again, intelligence – in the form of predictive models of event occurrence and patterns in the resulting loading – may provide a key piece in addressing this type of disruption.

distributed processing and storage resources: A trend with the potential to revolutionize high performance processing is the notion of "any time, any where" computing. In this model, the user is not aware of where, within highly interconnected web or network of computing and storage facilities, his job will actually execute or his data will be stored – a notion that, in this context, is sometimes called the *virtual archive*. In support of this capability, an *internet-scale operating system* moves the job or data to the best location based on various performance, reliability, and cost factors[2]. In the context of our problem, the archive might conceivably have access to a variety of remote processing and/or storage facilities, and itself might be subject to additional loading as remote jobs are dynamically assigned to its own resources. As one example, when a product requires as input data from three separate sensors, and when that data is geographically distributed across multiple archival locations, the problem of staging the data to the place where the processing will occur (at the archive, or at some other remote facility) is demanding. To solve it will require system-wide models of current loading and latencies, and a distributed optimization approach (*e.g.*, based on auction or adaptive control approaches). Such capabilities require modeling and intelligent optimization that go well beyond current practice.

use of priority and value in processing, cache, and request queue management: Concurrent demand for constrained resources is a classic problem in optimization. The system must adjust the order of processing both to maximize throughput and to increase value correlated to latency. In addition, it is possible that some users have (say, by policy, or through funding) higher priority than others. In this context, the archive may be expected to routinely adjust its processing flows and resource allocation to optimize *total value*.

increasing system complexity and labor costs: While the cost of computing, storage, and communications capacity continues to decline at a rapid rate, we cannot conclude that archives of the future will at some point be free. The complexity of complete archive systems continues to increase, driving up the cost of personnel to manage archive operations and potentially leading to a point where problem solution is so complicated and time consuming that data availability goals suffer. Greater intelligence in the archive could help contain operations costs and increase the level of service provided.

CHARACTERISTICS OF AN INTELLIGENT DATA ARCHIVE

The purpose of this section is to summarize and briefly expand on the types of functionality that characterize an Intelligent Archive. We have identified six areas, or types of behavior, that together constitute our notion of *intelligence* in this context.

OPTIMIZATION OF HIGH PERFORMANCE DISK CACHE

The difference in retrieval latency for data on disk, as compared to data in the robotic tape server, is currently about 5 orders of magnitude: from 10^{-3} seconds for disk access, to 10^{2} seconds for tape access. Often, improving latency in product delivery hinges almost entirely on whether the desired data is already on disk. A similar issue arises when some portion of the required data resides at a remote facility, and must be retrieved by the local archive to support product formation. If the data necessary for production (and often multiple sensor types and data sets are involved) have already been staged to disk, a large part of the latency budget is eliminated. An intelligent archive, then, is one that reasons about when, and under what conditions, to move data sets from the tape archive to disk, and when (similarly) to delete a data set from disk because it is unlikely to be requested, or to make room for other data sets.

Hierarchical storage management systems currently attempt to optimize file access based on observed usage heuristics, such as recency or frequency of usage, and cost functions related primarily to the size of files, often in combination[3]. However, most such algorithms do not make use of any knowledge about the file contents. On the other hand, analysis and modeling of past usage patterns and data requests in response to known types of event is one possible future strategy. Here, the archive knows (based, for example, on data mining of historical request log data) that certain kinds of event will (with high probability) result in identifiable and predictable data request and usage patterns for certain kinds of data. When the event is predicted – perhaps using predictive geophysical models – or detected, the IA can begin the process of pre-staging the data to disk – both from its own tape servers and also from distributed locations. The result is that, when the anticipated requests do arrive, the data to service them will already have been placed in the disk cache, thereby accelerating processing and delivery.

OPTIMIZED MANAGEMENT OF REQUEST QUEUES

There are a variety of optimization strategies based on analysis of the existing, or anticipated, request queue. For example, it might be valuable to postpone production and delivery to one customer for a bit, knowing (or predicting) the occurrence of other similar requests, and thereby permitting broadcast product distribution rather than multiple sequential individual deliveries. The result is overall throughput increase at the expense of small added latency to one customer.

It may also be that analysis of the processing or input data required to service one request partially overlaps that required for another. By grouping these requests and scheduling a single job that produces both products, it may be possible to reduce processing time and resource utilization. The ability to perform such optimizations requires explicit models for the required inputs and processing, and the ability to recognize partial redundancy. The algorithms to perform such analysis already exist (developed, in part, to support efficient compilers for parallel processing)[4].

In addition to latency and throughput *per se*, it is possible to optimize based on customer priority, and to recognize that the value of certain products decays with time. In order to "book" value for these requests, they must be moved to the head of the queue. And, since their value is evanescent, it likewise makes sense to purge them from disk at an early opportunity. For other requests, low latency is not an issue. By recognizing this, the archive can stage or postpone production, packing these jobs efficiently around high priority, low latency requests.

THE IA AS AN EFFICIENT REQUESTOR

The IA must be conceived as part of a connected network of storage and processing facilities, linked by an *internet-scale operating system* that allows access to remote resources. This considerably broadens the range of possible strategies available to the IA for servicing requests. In the simplest case, only locally available data and resources would be employed. At the complex end of the spectrum, the IA might choose to schedule a complex interdependent set of jobs and data transfers at and among a variety of remote sites. Here we meet again the notion of the IA as an *efficient requestor*, optimally coordinating the actions of a variety of service providers to achieve a complex result.

One critical component for solving this type of problem is a good set of models for the processing steps and the resources they consume: bandwidth, processor/memory, latency (= FLOPs), etc. Another is a distributed mechanism (e.g., auction algorithm) for negotiating an acceptable and/or optimized solution among the various service providers. (Some e-Business applications might provide an initial capability of this sort off-the-shelf.) Finally, the end result as perceived by the customer must be considered – that is, the optimization and/or feasibility constraints facing the IA as it balances the overall value-to-the-customer of services delivered.

PREDICTIVE AND ADAPTIVE MODELS

All of the characteristics of an IA so far discussed share a common feature that deserves to be broken out for separate discussion and emphasis: the need for reliable predictive models of events and processes. Here is a partial list of important models needed to support the types of desired capability.

- performance models associated with storage, networks, and processing
- models for data sets (internal structure, size, location, dependencies)
- models for job execution streams, including internal and external dependencies
- models for availability and performance of remote facilities and service providers
- models and stochastics for geophysical event occurrence (temporal, geographic)
- correlations among detectable events (tip-off, cuing)
- correlations of user request characteristics to the occurrence of geophysical events, to annual or diurnal cycles, or to each other
- orbital and sensor models for target accessibility
- models for product value and priority (e.g., value as a function of latency)
- models of feedback loop dynamics and pathologies resulting from event detection and sensor retasking (ephemeris, accessibility, downlinks, etc.)
- models embedded in algorithms for pattern recognition and classification (e.g. results of machine learning)

A state-of-the-art enterprise not only creates and utilizes such models – it also continually validates, revises, updates and adapts them based on actual observations from operations. The models are routinely compared against observations and system state estimates; and, if necessary, they are revised so as to ensure continued predictive accuracy. We refer to this process as *adaptive modeling* (not to be confused with *adaptive behavior* in response to event occurrence, discussed further below). An infrastructure to support state estimation for this purpose should be provided as part of the system design. The need for on-going validation and adaptation of supporting models, which is well-understood within the control theoretic community, is essential for an intelligent archive to reach its full potential.

EXPLICIT USE OF CONTENT-BASED METADATA

In resource optimization, an intelligent archive must be aware of the semantic content of the individual data sets that pass through it. There is often a strong correlation between the nature of the geophysical events recorded by (that is, contained within) a data set, on the one hand, and the frequency and type of use that will be made of that data set by the user community, on the other hand. We believe that an IA will routinely extract such content as part of ingest processing, produce the associated metadata, and use it to dynamically trigger adaptive behaviors: sensor retasking, selection among alternative data processing flows, identification of events correlated to known request and usage patterns, etc. Machine learning techniques may be able to provide computationally efficient pattern recognition and classification algorithms for this purpose.

This metadata may also serve as input to data mining algorithms intended to extract usage patterns and correlations from request logs. That is, it may be possible automatically to identify significant event-to-usage correlations, provided that the data mining algorithms have some knowledge of event characteristics via the metadata.

ADAPTIVE BEHAVIOR

Here, we refer to the ability of the archive to dynamically and autonomously adjust its behavior based upon the occurrence of events. The events may be geophysical; they may reflect changes in usage request patterns; they may reflect changes in the status of the local and remote resources available to the IA; or they may reflect changes in priority, value or latency requirements associated with products.
If optimization techniques are employed, they will necessarily be temporally local, based on predictions or estimates of event stochastics over some time horizon. As time moves forward, what had been a prediction (or estimate) now becomes known in fact, and the additional information can be used to re-optimize over a suitably extended new horizon. Such an approach, known as *rolling horizon* re-optimization, permits smooth transitions across event boundaries, since the optimal response is continually reintegrated over largely overlapping temporal windows. Thus, the optimized solutions adapt in a graceful manner as additional information becomes available and as statistical confidence improves.

EXAMPLE USAGE SCENARIO: FIRES IN THE WESTERN US

The following discussion illustrates several of the important themes in the preceding discussion within the context of a hypothetical scenario: support to Federal and State fire fighting and crisis management services responding to large fires in the Western United States.

Pre-season activity: A number of science-based predictive models are available to identify regions where fires are most likely to occur. These include: weather and climate models (rainfall, humidity, temperature); forecasts for vegetative growth; and forecasts for fire occurrence, density, and severity. Other models can be coupled with these to assess societal risk (e.g., housing near high-risk areas), and flood risk resulting from run-off from deforested areas. The IA has several roles and value-added functions during this period. First, it can collect historical data from disparate sources to support these models, pre-staging it well in advance of production to reduce user-perceived latency. Second, it can serve as an *efficient requestor* to coordinate production using these models, either applying its own resources, or negotiating with linked but distributed facilities, and optimally staging input data sources to them in a timely manner. Third, assuming that this process has been going on for several years, it can perform data mining on content-based metadata seeking past examples both to confirm/validate model predictions as well as to suggest possible correlations that might otherwise escape human attention. And fourth, it can serve as the mechanism for tasking/retasking high resolution sensors in areas perceived to be at greatest risk, thereby increasing model fidelity and forecast accuracy. The archive can become a value-added tool in the scientific modeling and forecasting process, simplifying the process of assembling and staging data, extracting knowledge through sophisticated exploitation algorithms,

searching for undiscovered patterns and correlations, and optimally utilizing other related computational and sensor resources.

During fire season – surveillance: Once fire season has begun, the IA begins a continuing process of event detection using content-based pattern recognition processing of ingested sensor data. This, in conjunction with other event detection capabilities, becomes the initial alert that a fire may be underway. This process of surveillance and event detection continues throughout the fire season, and is captured in the form of content-based metadata for later analysis. We envision use of computationally efficient machine-learning-based pattern recognition algorithms to perform the initial scan. When possible events occur, more sophisticated science-based models are automatically invoked (to reduce false positive errors). Inputs from precision weather and vegetation forecasts are used to update risk models, and the IA maintains data from areas at greatest risk in high-speed storage.

Responding to a fire event: When detection of an incipient (*i.e.*, nascent) fire event occurs, the IA can immediately task high resolution sensors to collect more detailed localized information. It invokes very low-latency data delivery paths to concerned organizations. Sensor constellation accessibility and diurnal periodicities are used to optimize collection and product delivery schedules. Models are invoked (perhaps at the archive itself) to assess severity and forecast temporal and geographic extent. This data can then be fused (perhaps at the archive itself) with other data sources to support GIS analysis and display. Flood analysis can also be performed, and data pre-staged for regions downstream of the fire area that may be at risk.

As the fire event proceeds: Under the assumption that the initial detection becomes a major fire, the IA can become the "data hub" for distributing low-latency, high-value data to a variety of Federal agencies. Daily progress summaries, fusing data from many sources, can be produced at the archive and disseminated in near-real-time. When more than one fire is active, the archive can supply data needed to assess relative severity and direct limited resources to areas that are most at risk. It can perform metadata-based data mining to identify similar past events, and use this to update models and validate forecasts. It can invoke models to monitor progress of Burned Area Emergency Rehabilitation (BAER) efforts, and it can use flood models to provide alerts for down-stream at-risk areas. It can monitor and predict smoke plume, pushing results to public health agencies. In short, the archive becomes a valuable resource for executing heavy-lifting models, staging and fusing data from multiple sources, providing value-added products to analysis and display systems at customer sites, continuing to provide event detect, alerts, and retasking services, and in general serving as the hub of a computationally complex and data-heavy analysis process.

Concurrent activities: In the meantime, independent of the fire scenario itself, the IA continues to perform its other duties and responsibilities. Intelligent planning and scheduling algorithms predict system load increases resulting from support to fire-related activities, and adaptively optimize ongoing production schedules. Data mining algorithms are invoked to analyze past events to predict system loading and latencies. This analysis serves as the basis for automatic negotiated work distribution across other linked processing and storage facilities. Feedback loop analysis and prediction is invoked to detect and avoid incipient pathologies (live-lock, dead-lock, hysteresis, chattering). High priority customers from other disciplines continue to receive high-quality, low-latency service despite increased system loading. Predictive models are routinely and automatically compared against observations for purposes of validation and/or adaptation. Management is presented with accurate, complete reports of current status, production schedules, predicted loading, disk cache policies, and assessment of value to end users. Worst-case what-if scenarios are extrapolated from current state, and optimized playbook entries prepared in advance for future selection and execution.

2. CANDIDATE TECHNOLOGIES

The following four subsections respectively address: technologies for managing the high performance disk cache, a key component in reducing latency of product delivery; technologies addressing problems in open-loop planning and scheduling of processing resources, including rolling horizon and mixed-initiative approaches; technologies supporting machine learning, pattern recognition, and automatic model generation; and finally, a brief discussion of *Model Predictive Control* and the associated control theoretic formulation of the problem and capabilities described in the prior section.

DISK CACHE MANAGEMENT

Access to data sets, both as input to production and for delivery to customers, is a source of latency. If the required data is already in the high performance disk cache, then this latency is greatly reduced when compared to the latency involved in robotic loading of cassettes, followed by a linear search along the tape to the point where the data resides, followed by a comparatively low-bandwidth data transfer. Thus, management of the contents of the disk cache is potentially a source for significant performance improvement (where performance here means primarily reduced latency in product formation and delivery). Even if all data were maintained on disk or memory in the future, the same logic can be applied to the management of local caches used to stage data from remote locations (esp., in an internet operating system scenario) for the purpose of reducing the cost or perceived latency associated with data retrieval.

A long-standing and successful simple rule for managing a cache is *LRU* (least recently used). When the system requests a data set to be loaded to the cache, the cache itself decides which data set to delete to make room for the new entry, and the rule it uses is based on maintaining a time history of usage: the data set whose most recent usage is the oldest is deleted first, and the process continues until enough room is made for the newcomer.

The difficulty with the rule is that it does not take into account such matters as the content of the data set, its potential relationship to other data (say, for formation of complex products), the fact that it may have been staged in anticipation of usage based on predictive models, and the fact that once a data set has been used, its value may almost immediately deteriorate (*e.g.*, highly perishable collection value). In other words, the true worth (that is, estimated value) of any particular possible loading of the cache is, in reality, a complex function of many factors, only one of which is customer interest as evidenced by recent usage.

In the discussion on Model Predictive Control below, we will suggest that cache management be integrated into a broader optimization approach that takes into account a variety of loading and value considerations. For the time being, it suffices to note that intelligence, in the form of models for value and stochastic estimates of collection, usage, and production patterns has the potential to improve cache management *vis a vis* current practice. These improvements are likely to be of greater value as the collection environment becomes more dynamic and reduced latency assumes increased importance.

OPEN-LOOP OPTIMIZATION

In this section, we will discuss three approaches to optimizing performance of the IA: schedulers (short-term horizons, on the order of hours); planners (long-term horizons, on the order of days, weeks, or months); and play-books (mixed-initiative approaches facilitating rapid review and customization by operators). A fourth section discusses issues of a rolling horizon and real-time, event-driven re-optimization.

These solutions have been characterized as *open-loop* because they do not explicitly model their own presence in the system. When we discuss Model Predictive Control below, we will take up a powerful alternative

approach to optimization based on closed-loop, self-referential models drawn from recent advances in control theory.

SCHEDULERS

At this level of abstraction and time granularity, we are considering issues of near-real-time assignment of jobs to processing resources and/or service providers. What jobs will be accomplished over, say, the next six hours; what processing, communications and storage devices will be used to execute those jobs; and what alternative contingency options are provided in the event of the occurrence of unscheduled or uncertain events? When we consider the IA as an *efficient requestor* of services, it is the scheduling function of which we speak.

The simplest approach is to build canned scripts to describe archive behaviors, negotiate firm interface agreements with external sources/consumers and simply let the machine run. This is simple and effective when day-to-day processing flows are fairly stable, or when a small fixed number of processing flows can be defined in advance, the duty of the system being simply to execute the fixed schedule chosen for it.

A much more difficult approach, but one with the potential for significant advantages in flexibility and customer service, is to dynamically schedule resources based on solving a complex optimization problem, where the objective function to be optimized reflects policy and resource constraints as well as measures of performance and customer value. Depending on the form in which this problem is posed (e.g., the form of the objective function and constraints, and the mathematical formulation of the solution space), a number of solution techniques may be applicable, including: linear programming; integer programming; stochastic search (e.g., genetic programming, simulated annealing, global integral optimization); and dynamic programming. Significant engineering tradeoffs are then required to balance quality of solution against the time and resources needed to calculate the solution. Further, the models embedded in a solution may themselves need to be adapted over time based on observed system characteristics and behaviors (see the more detailed discussion in Section 3 below).

A schedule produced by an optimizing scheduler has a period of time during which it is valid, as well as assumptions about expected system behavior and anticipated loading which, if violated, will render the solution sub-optimal or infeasible, and thus require rescheduling or re-optimization. The likelihood of the occurrence of events triggering rescheduling (either *de novo*, or as an adjustment to the current published schedule) is a significant factor affecting the engineering of the scheduler. The more dynamic the system and its environment, the greater attention must be paid to flexibility, adaptability, and robustness. Along this spectrum, it appears that current operations are very stable and loading is highly predictable. One of the drivers toward intelligent schedulers is an anticipated shift toward increasingly more dynamic environments.

Finally, we observe that scheduling may be a distributed function, in that the schedule adopted by any given facility may depend, in part, on the shift of some loading to other facilities, as well as the acceptance of additional loading from peers. Thus, the scheduling algorithm executed at the IA (conceived of as a stand-alone processing facility) may be part of a more general distributed optimization spread across and touching many other facilities. The management of the scheduling process itself can thus become a challenging problem, and models of the uncertainties of contingent dependence on other facilities take on increased importance in a world of virtual archives and distributed processing.

PLANNERS

Here, the time horizon stretches out to days, weeks, or months, and the local "peaks" of loading are smoothed away by suitable abstraction and statistical aggregation. In a distributed environment, loads are conceived of as system-wide, and the job of the planner is to raise confidence that the total load can be temporally and/or geographically leveled while still meeting system latency and throughput requirements at

scheduler-level time scales. The models that are of interest concern processes that may extend over a considerable period of time – e.g., seasonally correlated events such as fires, weather phenomena, and crop cycles. These geophysical models can be used to predict (stochastically) increases in certain types of loads and associated product requirements. Another important source of models is data mining of historical usage logs, searching for trends and event-to-usage stochastics, correlations and patterns. Similarly, the predicted system behavior embedded in plans can be tracked against observed behavior, flagging points at which actual loads begin to exceed anticipated thresholds, and thereby triggering plan adjustments or re-plans.

Unlike schedulers, which ordinarily pose a problem as a constrained optimization, planners tend to search for *satisficing* solutions, *i.e.*, an approach that meets all the goals and constraints, rather than a full optimization over a large search space. Because there may be many satisficing solutions, human review and tailoring is often associated with plan generation software. The extent to which planning of this sort can be fully automated (or will be automatable in the future) is open to debate. It seems reasonable to assume that some level of human review and approval will often be required, and that out-of-spec behaviors will trigger alerts to archive operators and managers (see the discussion on mixed initiative below). This approach also fits with latency requirements. Planners tend to have considerably longer to execute than do schedulers, which may be required to rapidly re-optimize in response to uncertain events. Intelligence in the IA can support the planning process through model generation and maintenance, data mining, software to propose plans for review, and automatic alerts when observed system loading or observed behavior strays away from planned predictions or assumptions.

MIXED INITIATIVE CAPABILITIES, AND PLAYBOOKS

These are approaches in which there is shared responsibility between decision and control software, on the one hand, and human managers and operators, on the other hand. The relative portion of the job assigned to one or the other of these resources (human and machine) must be *adjustable* based, for example, on response latency requirements. When there is plenty of leisure (deadlines are remote), the human can be proportionately more involved than when (by contrast) decision time lines are very short and latency is a paramount concern. Engineering this trade-off is difficult, but there are a variety of AI techniques now available or in development that support adjustable autonomy of this sort.

A particularly compelling example is the notion of a *playbook*. Based on a football metaphor, the playbook consists of a predefined set of schedule templates prepared as possible responses to anticipated kinds of event or disruption. By *template*, we mean that not all aspects of the schedule are fully specified. The system has the ability to suggest ways of filling out and completing the schedule, but the human can also take control of this process, and can modify or tailor the schedule to suit current goals via a specialized editor which supports play/schedule visualization and graphical interaction. The software for this type of schedule editing is powerful and adaptive; over time, user styles, preferences and tendencies can be learned by the editor for purposes of user-specific customization.

A concept of operations based on the use of playbook might include an ongoing "what if" simulation capability interacting with the planners and schedulers. When the simulation generates plausible event sequences that strain or break the system, a play (or plays) to address the situation can be developed and added to the playbook. While this capability is, today, a manual process, one can imagine automating it, with an off-line simulation and analysis engine searching for pathological input sequences, analyzing their plausibility, and automatically generating system response templates for entry into the playbook.

ROLLING HORIZON TECHNIQUES

As discussed above, both schedulers and planners typically operate over set time horizons suitable to the problems they are trying to address. They use models to make loading or event occurrence predictions over the time period of interest, and then produce feasible and/or optimal solutions of the associated constrained

problem. In a typical concept of operations, the schedules (or plans) are then published and used – either until the horizon has expired, or until some event occurs that renders the schedule infeasible or severely and obviously suboptimal.

Another strategy, however, is to solve the scheduling problem over a time horizon, say $[t_0, t_0+\Delta]$, but with the intention of resolving the problem anew at some intermediate time, t_1, where $t_0 < t_1 < t_0+\Delta$. The new solution would be solved over a new, extended horizon $[t_1, t_1+\Delta]$. This is then repeated for t_2, with $t_1 < t_2 < t_1+\Delta$ over a new horizon $[t_2, t_2+\Delta]$, and so on.

While it may be objected that time is wasted in generating the schedule over the unused interval $[t_1, t_0+\Delta]$, often the effort can be reused, since for many algorithms (e.g., simulated annealing, or genetic algorithms] the previous solution can serve as an optimized "seed" to begin the new search process. Further, the length of the optimization horizon induces an inertial, smoothing effect that keeps the system from settling too quickly into short-term local extrema[5].

To make such a scheme work, there must be available models to predict how the system will behave over the time horizon of interest. Since significant events may be stochastic, this means somehow sampling the space of possible futures, and arriving at schedules (or plans) that optimize, say, expected value, or that maintain a high probability of staying away from undesirable breakage (that is, measures of robustness are used as the objective function). When we use the word "intelligence" of an IA, it is the formation, maintenance, and algorithmic use of such models that we principally have in mind. One advantage of rolling horizon approaches is that they use new information that has been collected in the elapsed interval $[t_0, t_1]$ to update the models, thereby also improving prediction accuracy over the overlap $[t_1, t_0+\Delta]$ as well as extending the prediction to the new interval $[t_0+\Delta, t_1+\Delta]$. Thus, as new information becomes available, it is automatically and systematically used to improve model accuracy and, hence, associated optimizations[6,7].

In practice, it is not necessary to wait until the selected new schedule time, t_1, to perform re-optimization. If an event occurs at, say, time t_e prior to t_1 that jeopardizes the schedule calculated at t_0, there is nothing to prevent throwing out the existing schedule, and recalculating *de novo* a new schedule starting at t_e that takes account of the event.

Rolling horizon approaches are well known and used for modeling of physical and geophysical phenomena (e.g., weather forecasting). There is no reason why they cannot be equally effectively applied to management of processing flow and data staging in a processing facility, like a data archive, that is itself subject to stochastic events with controllable internal responses.

MACHINE LEARNING, STATISTICAL ANALYSIS, AND DATA MINING

In this section, we will briefly mention several modeling approaches, and their potential applicability to an Intelligent Archive.

Pattern recognition and classifiers: In a typical application, training data is prepared using skilled human beings, past observations, or highly accurate model-based scientific codes. These training sets are then used to generate efficient codes that reproduce, on an input-output basis, the results produced by the high fidelity classifier. The advantage is that these new algorithms are very fast, in effect trading off computational complexity against acceptable (based on practical considerations) Type I and Type II error rates. There are a large number of such techniques, including (most prominently) *Neural Networks* (of various flavors), *fuzzy logic, and rule-based classifiers* (perhaps derived automatically as the result of genetic algorithms searching through rule-space). Most recently, there has been much interest in *Support Vector Machines*[8], in which the natural domain (represented as a vector space) is mapped by an inner-product-preserving transformation onto a Hilbert Space, whereupon the training set is optimally

separated via a hyperplane. The primary utility for these algorithms appears to be in content-based extraction of events and/or metadata on ingested data and subsequent down-stream products. However, they could also be used in data mining of event and request logs to extract and learn patterns and correlations of system usage and loading.

Clustering: When training sets are not available, machine learning tools must boot-strap themselves based on the raw data itself. One way this is done is by searching for clusters of data in some suitably constructed metric space tied to the data. These clusters may, it is hoped, reflect underlying interesting structures or correlations. For example, a cluster taken from request logs might identify, in a suitable vector representation, geographical and spectral correlations. This, in turn, might suggest to the analyst the occurrence of a type of geophysical event. The occurrence of this event, in the future, might then be used to predict or suggest the associated type of loading patterns, allowing the IA to pre-stage the data to disk and provide improved latency of product delivery. Clustering is a widely used tool in data mining packages and applications[9,10].

Statistical analysis and regression: While not strictly speaking "artificial intelligence," statistical analysis of data sets is a widely used and very effective way of extracting useful information and summary characterizations of large data sets. It is almost certain that an intelligent archive would make use of such techniques, both as a way of data reduction and generation of metadata, and as a means for building and maintaining accurate stochastic and/or probabilistic models of geophysical quantities, as well as the relationship between data content and usage.

Entropy-based rule induction: Another boot-strapping technique involves successively partitioning the underlying data set so that each successive division minimizes the entropy of the resulting partition (entropy is conceived here as an indication of information content). Either driven by a graphical user interface, or performed automatically, the rules in question amount to specifications of data separation hyperplanes. The end product of a sequence of such partitions may identify significantly correlated samples. A heuristic examination of these subsets may then yield insight into underlying similarities, structure or causality. The rule sequence can then be imposed on new data sets to identify the existence and extent of the associated characteristics. Again, the most likely place for application of such techniques is data mining of usage and request logs tied to metadata characterizations of the requested data sets[7,8].

Bayesian Networks: This well-known and widely-used technique formally characterizes internal correlations and conditional correlations in a tree-based representation. They can serve as the basis for Markov Chain representations of state space dynamics, since they can be interpreted as providing state transition probabilities. These Markov Chains, in turn, can support computationally efficient Monte Carlo characterizations of complex system interactions – a mechanism for modeling the stochastics and distributions associated with event-driven system evolution. As we have seen, these predictive models provide a very useful input to optimization algorithms attempting to maximize expected value, reduce maximum risk, etc.[11,12]

Fuzzy and Neuro-Fuzzy Modeling: As a final example, we refer to the impressive results that have been achieved through the application of so-called *ANFIS* (*adaptive neuro fuzzy inference system*) to a variety of linear and non-linear modeling problems. The message here is that the developer has a wide variety of approaches from which to select, many of which have a deep theoretical foundation. For example, ANFIS has been shown to have its own *Stone-Weierstrauss Theorem* – a guarantee of arbitrarily close approximation on compact sets[13].

MODEL PREDICTIVE CONTROL

In this section, we will consider a novel application taken from control theory to the problem of optimizing operations of an Intelligent Archive – *Model Predictive Control*. We will see that this approach offers an attractive unifying architecture and uses the substantial body of theory and ongoing research from the control theoretic community. For the interested reader, an Appendix has been provided to introduce MPC and discuss some of its potential advantages and challenges. Briefly, MPC uses predictive models of system stochastics and dynamics to estimate the effect of various control signals under consideration. It then applies an objective function to these predicted outcomes, and selects the control signal that achieves the best result (as measured by the OF). One strength of MPC is that it includes a model of its own behavior and ability to respond to uncertain events (that is, it is *dosed loop* in the classic, self-referential sense of control theory). Using the generic Figure 3 as a template, the discussion below will briefly consider: (a) the *control signals* an MPC controller might generate; (b) the types of *models* it will require; (c) *state estimation* (SE in the figure) for the models and entities in the system; (d) the form of the *objective function* (OF in the figure) and *optimizing algorithm* it might employ; and finally (e) a *concept of operations* for an MPC implementation tied to the illustrative scenario described in the section above.

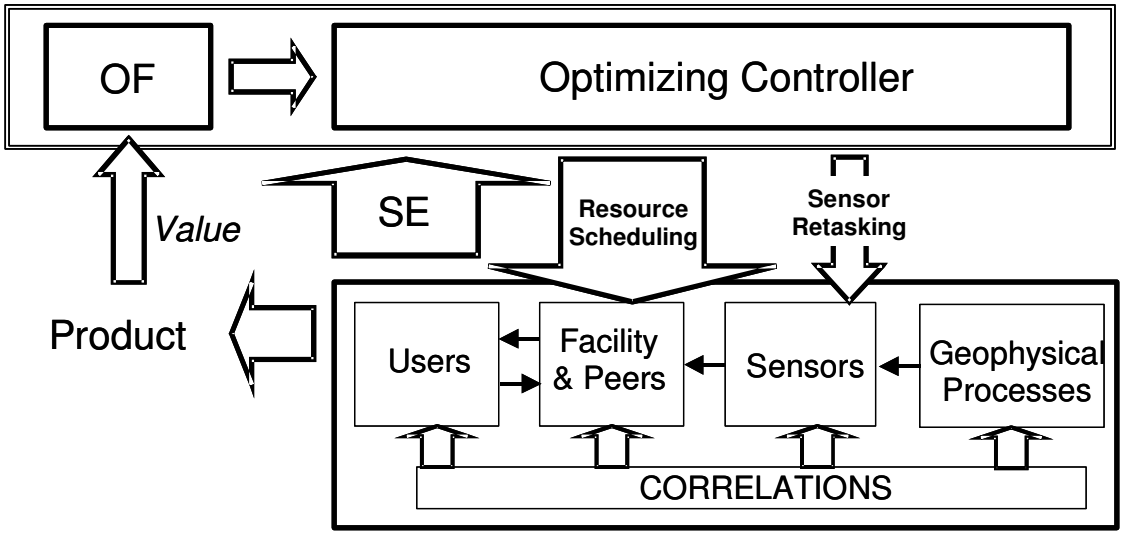

Figure 3. An MPC System Diagram for an Intelligent Archive. OF indicates the Objective Function used to evaluate the level of optimization. SE is the State Estimation, or information about the current status and performance of various elements within the system.

CONTROL SIGNALS

The resource controls the MPC can command or influence include:

- Hardware resource management – the disk cache, processors, robotic tape archive, and communications gear and bandwidth.
- Software scheduling – in particular, the selection of what algorithms to apply to incoming data streams; metadata extraction; process branching based on event detection; support for state estimation (model maintenance and adaptation, data mining for trending, clustering, usage pattern recognition); and optimization of MPC control decisions.
- Event generation – that is, tip-offs and sensor retasking requests in response to event detection.
- Product delivery – that is, the decision of when and how to send requested products to customers.
- Negotiation for services with distributed storage and processing peers.

In Figure 3, we have grouped the above into two major types of control signal: commands to the archival facility resources and peers; and re-tasking commands to the data providers.

MODELS

Models for the MPC controller fall into four groups (reference is made to Figure 1-1): (1) *geophysical processes* being observed; (2) the *data providers* – that is, the vehicles, sensors, and processing facilities that generate the input data stream; (3) the *IA facility* and its peers; and (4) the *user community*. In addition, an MPC controller for the IA will be interested in discovering and modeling (5) *significant correlations* between events occurring in each of these domains.

Geophysical processes: This includes predictive models of event occurrence known to require specialized processing, low-latency product delivery, additional loading, or other kinds of impact to the IA. The idea is that by understanding the likelihood of occurrence of such events, the IA can plan for them, and be better prepared to respond when they occur.

Data providers: This includes ephemeris and satellite constellation information, accessibility windows, types of sensing (and constraints – e.g., weather), downlink times, raw data processing and delivery latencies, retasking options and latencies, and retasking priorities. In addition to modeling loading (especially event-driven loading deriving from correlations with geophysical events), the IA of the future will need to understand retasking options, and the associated down-stream loading.

Facility and peers: This includes predictive models for performance (FLOPS, bandwidth, storage), error rates, processing flows, alternative processing strategies, and similar capabilities and demands from peer facilities.

Users: This includes models for both standing and *ad hoc* request patterns and associated loading (for example, seasonal periodicities reflecting academic or research calendars). It also includes significant event correlations and associated request patterns (see correlations below).

Correlations: A major goal is to understand how events in one area (geophysical events, for example) affect behavior in other areas (user requests, data provider tasking strategies, specialized processing and event detection, etc.).

STATE ESTIMATION

State estimation means the ability to assess and predict values and status of the various entities constituting the *plant* (that is, the system being controlled, and its environment), including the complex correlations and dependencies we discussed at some length in Section 1. Regarding the IA facility, the MPC controller must know the status of the disk cache, the job execution status and queue of the processing and communications gear; the status of negotiated loading agreements with network storage and processing peers; and sufficient information to stage data and sequence processing to achieve production goals. Concerning the sensors and collection environment, it must understand the accessibility patterns and downlink schedules, event occurrence stochastics, retasking policies in response to events, and event-retask-processing feedback loops. Concerning users, it must not only understand the current set of requests, but have predictive models concerning how users are likely to respond to the occurrence of geophysical events as well as cyclic usage patterns.

OBJECTIVE FUNCTION AND OPTIMIZING ALGORITHMS

The objective function captures management goals or rules for system operation. A more detailed discussion of the technical challenges facing definition of the **OF** is provided below. Experience has shown that the

definition of the **OF**, and the selection of relative value for parameters embedded in it, can be a source of significant discussion. Here is where latency-related product value is assessed, relative priority among users is adjudicated, alternative strategies for satisfying requests are compared and evaluated, and probabilities of occurrence are factored into overall performance objectives. Also, these issues are of direct concern both to users and to the IA management, and different organizational entities often have differing priorities for system behavior. Finally, the mathematical form of the **OF** can strongly affect the type of algorithm that can be efficiently employed to optimize it.

CONCEPT OF OPERATIONS

Armed with these models and an ongoing stream of **SE** inputs, the MPC controller is well positioned to perform any or all of the complex optimizations and adaptations we have previously described. Since it understands stochastics of geophysical event occurrence, it can anticipate the likelihood of such events, and position itself (in advance) to optimally respond (retasking, event detection, metadata, low latency delivery pipes, etc). Since it understands the correlations between event occurrence and user request patterns, it can anticipate expected loading, negotiate peer support, pre-stage associated data sets, and in general optimize for reduced latency and product quality. Since it understands its own resources and has control over them, it can arrange processing sequences so as to optimally balance routine standing processing against transient high-value low-latency event-driven loading, smoothing and balancing system loading over time horizons of sufficient length to prevent chattering and instability.

Recalling the example fire scenario from Section 1 above, it should now be clear how the MPC controller might be able to achieve the kinds of system functional requirements that were described there. All of the types of predictive behaviors – time and location likelihoods for fire occurrence, sensor retasking options and latencies, surveillance and event detection processing, user request patterns and associated loadings, etc. – have a natural "home" in the MPC framework. In short, *any phenomena felt worthy of modeling based on possible impact on performance can, and should, have a place in the MPC paradigm*. From a theoretical point of view, the approach is clean and satisfying: model both state and dynamics of what's important to sufficient detail to determine interactions with other components over time horizons adequate for smoothing and stability. Then, use these models to select among control options those that optimize selected performance goals (as captured in a formal objective function). And, do this in a way that takes explicit cognizance of the presence of the controller and its own ability to respond and adjust to dynamic and uncertain event occurrence.

3. TECHNICAL CHALLENGES

This section addresses the most significant technical challenges to optimizing resources in an Intelligent Archive. Seven subsections discuss, in turn: *model* construction, monitoring, and adaptation; definition of a suitable *objective function*; *computational complexity*; feedback loop *pathologies*; *state estimation*; *hybrid systems*; and *hierarchical and distributed architectures*.

CONSTRUCTION, MONITORING, AND ADAPTATION OF MODELS

Building models suitable for practical use is something of a cottage industry for applied mathematicians. It tends to be expensive, and the models tend to be brittle. Self-constructing, data-driven models, by comparison, have the potential to automate quite a lot the model construction process. A neural net, for example, needs only a well-constructed training set and an interface to the data to produce, in many instances, quite satisfactory pattern recognition or classification routines – two of the capabilities we have identified as needed for the IA (to support metadata construction and event detection, for example). In any case, for the IA to be practicable from both cost and adaptability perspectives, means must be provided for automatically generating, monitoring, and adapting most or all of the models used to control the system.

The most promising approaches currently on the horizon are various types of machine learning and database mining. For example, good training sets can be derived using high-precision (and computationally expensive) science-based algorithms; these can then be used to train more computationally efficient ML instantiations to meet the less exacting requirements of operations optimization. Likewise, database mining of history logs focused on patterns and correlations in event occurrence and user request patterns can identify significant trends and system dynamics. Other models must capture job sequencing and expected performance, disk cache behaviors (correlated to usage patterns), negotiated services from processing and performance peers, expected latencies of sensors and downlinks, and user priority profiles and latency requirements.

Concerning adaptation, predictive approaches have the advantage that *predicted* system behavior is continually available for comparison against *observed* system behavior (see the discussion below on state estimation). The difference (in the appropriate space) between the two (the *residuals*, to use the terminology of Kalman Filters) becomes important information both to signal the occurrence of events and/or as input for model adaptation. Dynamical models can be constructed to automatically produce and use this type of residual information[14,15,16,17,18].

System integration of models will also be a challenge. None of these models, in and of itself, is beyond the state of the art. What is a stretch, by today's standards, is the notion of integrating all of these models into a global predictive model of system behavior and stochastics adequate to support even short-term optimization decisions. The alternative, however, is a piece-meal collection of separate controllers and optimizers, each focused narrowly on one aspect of the problem, with no overarching, system-level view to balance among them and detect/prevent pathologies. In such a system-level approach, adaptation and insertion of improved models will require strict interface definitions and well-defined processes for validation of model performance (accuracy, speed, robustness, etc.)[19,20,21].

DEFINITION OF AN OBJECTIVE FUNCTION

In a fully integrated approach, the objective function is tied to top-level measures of performance (such as "value delivered to customer"). Such a measure necessarily aggregates many other component or surrogate metrics, but must accurately capture user and management assessment of system performance. The key components that might enter into such a top-level value function include:

- *Latency*: primarily event-to-delivery latency, to which the archive contributes;
- *Product quality*: particularly any value-added processing, fusion, or registration that might have been provided;
- *Adherence to published delivery schedule*: that is, meeting promised production timelines, so that the user can confidently plan;
- *Customer priority*: reflecting the agreed-to pecking order in the user community;
- *Product correlations*: that is, honoring the fact that some data requests are only of utility if others are also provided; and
- *Metadata services*: ease and precision of search.

Given these top-level measures, there are other internal secondary metrics, relatively invisible to the user, that measure efficiency. Notice that if improved efficiency leads to improvements in the above-described user-level metrics, then these secondary measures may not need to be separately and individually optimized. These other metrics include the following:

- *Resource utilization*: keeping as much of the system resources usefully engaged as possible;

- *Peer-to-peer negotiations:* off-loading processing in peak periods, and accepting additional processing in slack periods, for system-wide load-leveling;
- *Throughput:* gross measures of over-all amounts of product delivered, irrespective of quality, priority and latency; and
- *Cost:* expenses for computation, storage, and communication that reflect the use of external and internal resources with varying cost/performance characteristics.

The mathematical form of the **OF** is a matter of concern, since it can impact the computational performance of the associated optimizing algorithms. Issues include:

- continuous dependence on key input variables
- convexity
- dynamic range
- level-to-level consistency in a hierarchical or distributed implementation

Finally, it must be possible to adjust the **OF**, more or less in real time, to reflect changing management priorities, as it is the most natural means for management to adjust system behavior and performance. One useful piece of information to feed back to management is the anticipated effect resulting from a proposed change to an **OF** parameter. It may even be possible for management to specify its goals on an input-output basis, leaving the system to automatically adjust **OF** parameters to achieve the desired behaviors.

COMPUTATIONAL COMPLEXITY

While the performance objectives above require a large amount of computing power in today's terms, the computational complexity should not be a significant obstacle in the long run. Two key elements that support this provisional thesis are: (1) Moore's Law, and (2) the inherent and natural parallelizability of the proposed approach (stochastic search). Perhaps the most demanding scenario involves use of an event driven simulation embedded in the recursive structure. (In formal terms, we replace the full backwards search called from by Bellman's equation with a "cost to go" estimate using the current state[22,23,24].)

Several techniques can accelerate or improve stochastic search. One that may be particularly relevant here is estimation of the solution *backbone*[25,26,27]. In a situation where value is dominated by a few key drivers, one imagines first solving a reduced problem focused solely on these "tall poles." The best (one, or a few) are then used as fixed seeds around which additional lower valued activities are tightly packed. It may also be possible to solve the reduced problem using classical techniques (linear programming [LP], dynamic programming [DP], etc.)[28,29] and use this is a high-quality initial solution to seed the stochastic search.

FEEDBACK LOOP PATHOLOGIES, STABILITY, AND ROBUSTNESS

An advantage of a control theoretic view of the problem of operations management is a ready-made conceptual context for many of the issues that arise in dynamical systems, together with a long research history of solution approaches. This context highlights the potential risk from feedback loop pathologies. Systems that are agile (that is, low-latency event response) can exhibit non-intuitive behaviors when they become self-referential (that is, when outputs are linked back, perhaps after several intermediate steps, to input). Examples of such pathologies include: live-and dead-lock (a system waiting for itself); chattering (rapid alternation across discretization thresholds); and ringing (positive eigenvalues lead to exponential power growth)[30]. Systems that are not agile, by comparison, may be so slow to respond (and, hence, so sub-optimal) that these behaviors are simply not an issue.

The *stability* of the system has practical consequences in the *plan-to-plan variability*. Heuristically, we seek to avoid situations where the optimizing controller produces very different plans (as perceived from the users' point of view) in response to very small changes in input. It is not unusual, for example, to find many feasible solutions that produce almost identical outputs as measured by the objective function, but are far apart when measured by a metric comparing plan similarity. The solution requires a trade-off that takes the operational impact of such variability into account, and is resistant to radical plan variations that achieve only marginal improvements in the **OF**. Both LP (linear programming) and DP (dynamic programming) solvers are notorious for exhibiting this type of behavior (that is, very small perturbations of the input conditions can lead to solutions that are virtually identical in value but that are dissimilar in selected operations and their order). Some algorithms (e.g., stochastic search, such as simulated annealing) have the property that they can accept the current plan as a seed for generation of the next plan. If a metric of plan-similarity is available, it may even be possible to build this directly into the **OF** itself [5,16,31].

System *robustness* refers to tolerance for errors in input state estimation and unmodeled dynamics. That is, a robust system can continue to maintain reasonable (though perhaps sub-optimal) behavior even in the face of uncertainty concerning current and/or predicted future system states. In practical terms, it means making continuous estimates of uncertainty and alerting operations personnel when thresholds have been exceeded. Thus, the system must contain models or estimates of its own confidence in its ability to correctly (within stated performance thresholds) continue autonomous operations. Perhaps the single most powerful tool addressing robustness is to make the system *adaptive*: that is, to incorporate estimation techniques capable of automatically tracking and adjusting parameters embedded in (that is, treated as mathematical constants in) dynamical models. One especially important example is parametric characterization of probabilistic or stochastic processes[15,16,32,33].

As new models and algorithms are introduced into the system over time, a rapid, repeatable process must be in place to validate that their incorporation does not compromise stability or robustness and does not lead to unwanted pathologies. Thus, the tools, tests, and adjustment procedures to initialize the system at its inception must become part of its ongoing ancillary support services. In effect, we must have the ability to rebuild the system on-the-fly in response to new capabilities or changes in the operating environment. Procedures and tools to "fix" the system and revalidate it should be (to the extent possible) automated and automatically initiated.

STATE ESTIMATION

An advantage of a control theoretic approach to this problem is that control theory (as the result of long experience) always and necessarily incorporates a full treatment of the state estimation techniques on which it relies[7,14,19,34,35]. As described in §2 above, there are five major groups of models for which SE inputs will be required: (1) geophysical processes; (2) the data providers; (3) the IA facility and its peers; (4) the user community; and (5) significant correlations among these four entities. Of these, (1) and (3) are relatively well-understood in the sense that models and supporting **SE** inputs in these areas exist and are currently routinely utilized for a variety of purposes. Concerning the data providers (2), models for the physics of orbital constellations, accessibility, and sensor characteristics are available. Less well-understood are the organizational and human aspects of sensor tasking and priorities. A similar problem exists in (4), where reliable models for user behaviors and request patterns (the understanding of which is critical to predictive optimizations) are not available. How does one "instrument" the user community in such a way that system loading and latency requirements can be anticipated and optimized? One promising approach is to use models of correlations (5) as a way to make these "hidden variables" observable (in the control theoretic sense). Thus, for example, occurrence of a geophysical event becomes a predictor of certain types of user request patterns and associated processing loads. It is precisely this linkage between observable events, on the one hand, and hidden characteristics, on the other hand, that motivates the development and maintenance of the correlation models. The log of system loading, performance, and user requests becomes a major

source of data for this sort of predictive and adaptive analysis. Data mining of this log data, *conceived as a form of state estimation*, will be a significant resource for system intelligence and adaptability.

An architecture incorporating state estimation considerations will contain two essential elements. First, a *physical infrastructure* is needed to gather and report timely state estimates to the models used by the optimizing controller. And second, *internal algorithms* must continually track the accuracy and reliability of the received data. The best such algorithms are based on *residuals* – that is, *predicted* state values are compared against the *observed* state values. Thresholds are established reflecting the operation's tolerance for uncertainty (and associated sub-optimality), and threshold excedence flags the need for adaptive adjustment (perhaps automatically performed, or perhaps with some level of operator involvement). As noted in the previous discussion, there is a tight coupling between the robustness of the system and the engineering associated with the state estimation function.

HYBRID SYSTEMS

In classical control theory, a *hybrid system* refers to a finite state machine (= discrete system) in which there is a separate controller (or control law) associated with each state. The idea is that as the system changes from state to state (reflecting major alterations in the external environment, for example), the system shifts to the control law (that is, the set of models and optimization approach) best suited to current conditions. In our previous discussions, this notion was to some extent subsumed in the notion of modeling fidelity and adaptation; select the model that best represents the observed state. Similarly, playbooks reflect a somewhat similar notion: select the play and tailor it to best match what is currently going on. Hybrid amalgams of continuous and discrete controls and techniques are widely used, and might well be appropriate to the intelligent archive[20,23,36]

DISTRIBUTED AND HIERARCHICAL ARCHITECTURES

Time horizons for the processes of concern span several orders of magnitude (from minutes, for production scheduling; to diurnal access patterns; to seasonal variations in geophysical processes). This fact tends to drive toward a hierarchical approach, in which higher levels in the tree correspond to models associated state estimates extending over longer periods of time; and in which processes with short response cycles reside at nodes lower in the tree. Maintaining control theoretic properties (optimality, stability, observability, controllability, robustness) across levels remains a challenging and active area of research[28,29,37].

We have also observed that an IA should be thought of as a single node in a distributed collection of processing and storage facilities. Thus, it is likely that some kinds of optimization will require peer-to-peer negotiations that converge iteratively to a global solution. There is a trade-off here against a single centralized solution. It may be that the final architecture computes some aspects of the solution using local/distributed techniques (e.g., for control nodes lower in hierarchy), and computes other aspects centrally (e.g., for nodes or processes over longer time horizons).

4. SUMMARY AND FUTURE DIRECTIONS

In this section, we will briefly recapitulate the major conclusions and indicate possible areas of emphasis for future research and development. The reader will have noticed an emphasis on a control theoretic view of this problem. The archive is conceived as one component of a larger system which includes the sensor constellation (and its management and tasking functions and facilities), the underlying geophysical processes of interest, the user community, and the archive itself. The archive has the ability to issue what we have conceptually termed control signals, both to its own internal processing and storage resources, but also (explicitly or implicitly) to peers in a distributed, networked environment and to other system components.

These control signals produce effects, which can then result in low-latency feedback loops – particularly when and if the system becomes more agile, flexible, and adaptive than is currently the case. A control theoretic approach models all the aspects of system behavior that might affect the optimality of the control decisions, and (we have suggested) does so predictively, both assessing future impact, and taking account of its own presence in the system to respond to future uncertain events if they arise. We have termed this approach to the problem *Model Predictive Control*, and noted that the controls community has already begun to use such techniques for optimized enterprise management – an application that until recently had been treated as an advanced problem in operations research. While MPC is not the only available approach to achieving the performance and optimization goals discussed in the problem statement, it provides a conceptual framework within which to discuss and assess all the major technical issues facing any proposed solution.

Three major drivers motivate the introduction of intelligence into archive. First, we wish to reduce processing latency by predictive pre-staging of data into the high performance disk cache. The predictive models that might be of use to do this include: temporal, geographic, and phenomenological models (e.g., of land-cover) providing stochastic indications of possible event occurrence (where, and when); user usage patterns, both event-based and as correlated to the cyclic nature of the academic and research calendar; and retasking feedback loops and associated specialized processing loads. Second, we wish to provide low latency tip-off and cuing for rapid sensor retasking. This requires defining low-latency processing paths both for surveillance monitoring and for focused high-precision localized analysis. This requirement overlaps and reinforces the need for more and better metadata, both to support as well as to support data mining for underlying usage patterns and correlations. The third driver is that the solution must exist within a distributed, interconnected network of processing and storage resources. This will require negotiation, and the ability to tradeoff network bandwidth and latency, on the one hand, against processing latency, system stability, and such future capabilities as on-demand-processing, on the other hand.

In addition to the MPC control theoretic paradigm, three other areas of technology are particularly relevant to this problem. First, there are the strategies for managing the disk cache – the algorithms that decide when to purge data from disk to make room for new data sets, and which data sets to choose. Secondly, there are resource optimization techniques such as open-loop planners and schedulers, as well as rolling horizon techniques that can smooth spikes and avoid temporally localized extrema. Finally, a number of machine learning and AI technologies are available to assist in the automatic generation and adaptive maintenance of the many models that will be required by any predictive optimization approach.

We also presented the significant technical challenge facing the development of such a capability. At the top of the list is the need for automated support for model generation and maintenance, including an architectural framework that recognizes this function as an integral (indeed, essential) capability. Another challenge is the definition of a suitable objective function, able to capture and balance the many competing demands and constraints on the system. A third is algorithmic and hardware approaches adequate to the computational complexity of such an optimizer. The difficulties of state estimation (and, in particular, uncertainty, adaptability and observability) and the pathologies of feedback systems were described. Another challenge is the use of hybrid systems – combinations of discrete state machines, with each state corresponding to a tailored control law or optimization approach. And finally, we observed that the eventual solution will almost certainly involve both hierarchical decomposition and distributed negotiation – areas that are well understood for some types of algorithms, but that are still research topics in the controls community.

All of these are currently active and supported areas of research in the control theory field. There is every reason to believe that, with continued support, all of the required technologies will be available within the time frame envisioned for this work. Since all will almost certainly be required for the eventual solution, we resist the temptation to rank or prioritize. That said, probably the area least understood at the moment is automatic model generation. While many machine learning algorithms have been proposed and analyzed, the

problem of integrating the right combination of these into a single, tightly coupled and autonomously adaptive system remains, at the moment, beyond our reach.

Appendix A - Introduction to Model Predictive Control

The purpose of this Appendix is to provide a brief, accessible introduction to Model Predictive Control (§A.1), and to raise some observations that should be taken into account in considering its use in an Intelligent Archive (§A.2).

A.1 Background and Definitions

In recent years, the controls community has taken up problems in *enterprise management*[6,38] – an application that had been considered special provenance of operations research and artificial intelligence (*i.e.*, providing decision support via database mining). When we apply a control theoretic view to an enterprise, something like Figure A-1 emerges. The *optimizing controller* selects among available, feasible options to send *control signals*, **CS**, to the *plant*. (The term *plant* derives from earlier days, when the application was (for example) a chemical processing plant or a refinery.) The selection of the character, timing, and intensity of these control signals is the purpose of the controller. Coming back to the controller from the plant are as series of *state estimates*, **SE**, reflecting both the state of the plant as regards settings, health and status; but also measurements of the physical underlying processes – in a chemical production plant, for example, these might include temperature and pressure, concentrations, and flow rates, as well as measurements of factors to some extent beyond the reach of the controller, but that have the ability affect product quality and timeliness. The controller uses these state estimates (perhaps in the form of actual measurements, but also in the form of derivative or predictive models, which may be stochastic) to optimize the *value* of the *product* delivered to the end customer. To this end, measurements of product quality are made and provided to an *objective function*, **OF**. The **OF** balances the many (often competing) measures of product value into a comprehensive assessment, using policy and directives from *management* for purposes of tuning and emphasis. Put simply, the job of the controller is to select feasible control signals (that is, control signals that satisfy all known policy and operational constraints) so as to optimize delivered value as measured by the objective function.

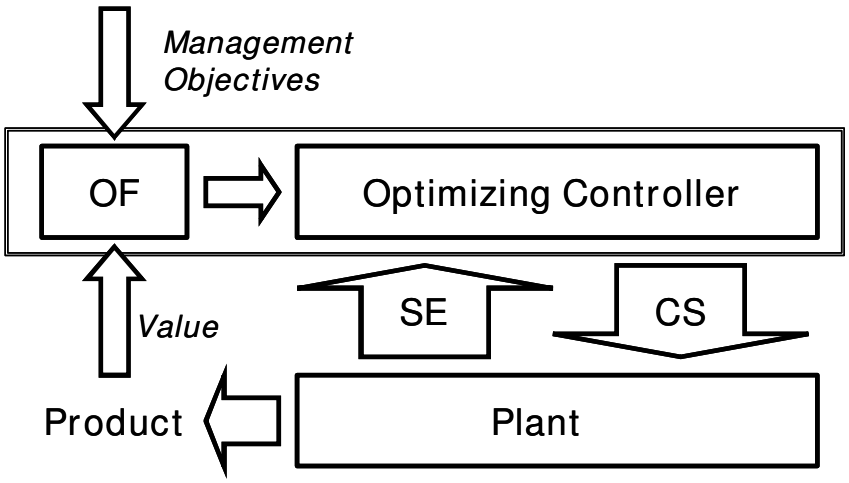

Figure A-1. A Control Theoretic View of Enterprise Management

The way in which a *model predictive controller* (MPC) approaches this problem is to attempt to predict what the effect will be of applying one or another of the available control signals to the portions of the plant under its control. The notional situation is shown in Figure A-2. In this simplified situation, confronted with three control options – C_1, C_2 and C_3 – the MPC plays a predictive model forward in time for each control options, producing corresponding estimated outputs o_1, o_2, and o_3. It then determines the best outcome, as

measured by the objective function, and the corresponding control signal is selected (i.e., if o_2 is the optimal estimated outcome, then C_2 is issued to the plant for execution).

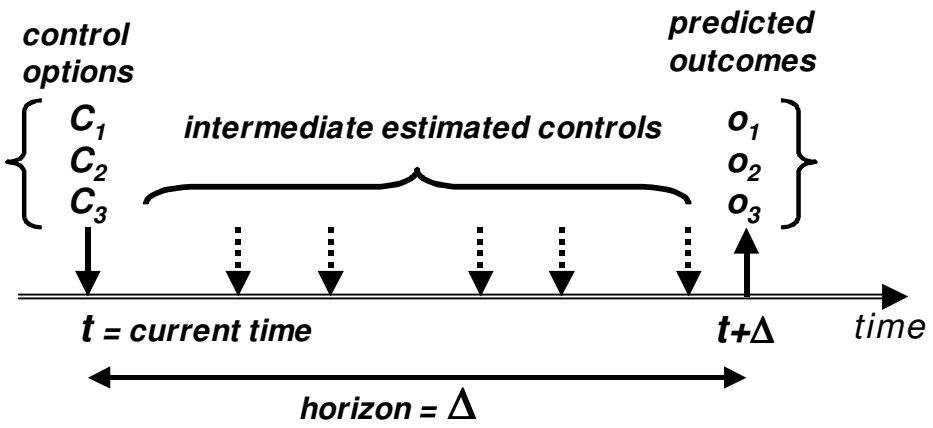

Figure A-2. MPC Optimizes Over a Time Horizon

Before proceeding, several observations must be made about this simplified representation. First, typically the MPC only considers a finite time horizon for its prediction – one consistent with the accuracy of the predictive models at its disposal. However, unlike an open-loop scheduler which produces a complete set of control signals covering the entire time horizon, the MPC only issues a control signal valid for a short period of time – much shorter than the length of the horizon being considered. The next point in time at which a control signal is required, whether scheduled or event driven, the process is repeated. Why, then, bother with the extended time horizon at all? The answer is that the MPC explicitly models the possible future control decisions that will have to be made, *and models itself as the optimizing entity making those decisions*. That is, the MPC is *closed-loop* (= self-referential) in that it explicitly models its own presence in the system, and its own ability to dynamically adjust to the uncertainties inherent in an underlying stochastic processes. Thus, the control signal actually issued by the MPC is the one that best optimizes long-range performance *under the assumption that an optimizing controller is present to respond to and re-optimize based on the occurrence of uncertain events*. The resulting forward-branching tree, with its stochastic properties, is the mathematical object of interest. If it can be represented as a discrete state space, Dynamic Programming becomes the solution method of choice [24, 25].

Returning briefly to Figure A-2, we can redisplay it showing how this predictive model capability is actually implemented (see Figure A-3). Within the MPC controller are models for all aspects of the system, including the MPC controller itself. Conceptually, the necessary and sufficient characteristic of these models is that they are adequate (explicitly, or implicitly) to support an event-driven-simulation of system response to the full range of possible system inputs, taking account of the presence of the MPC controller and the feedback loops in which it participates.

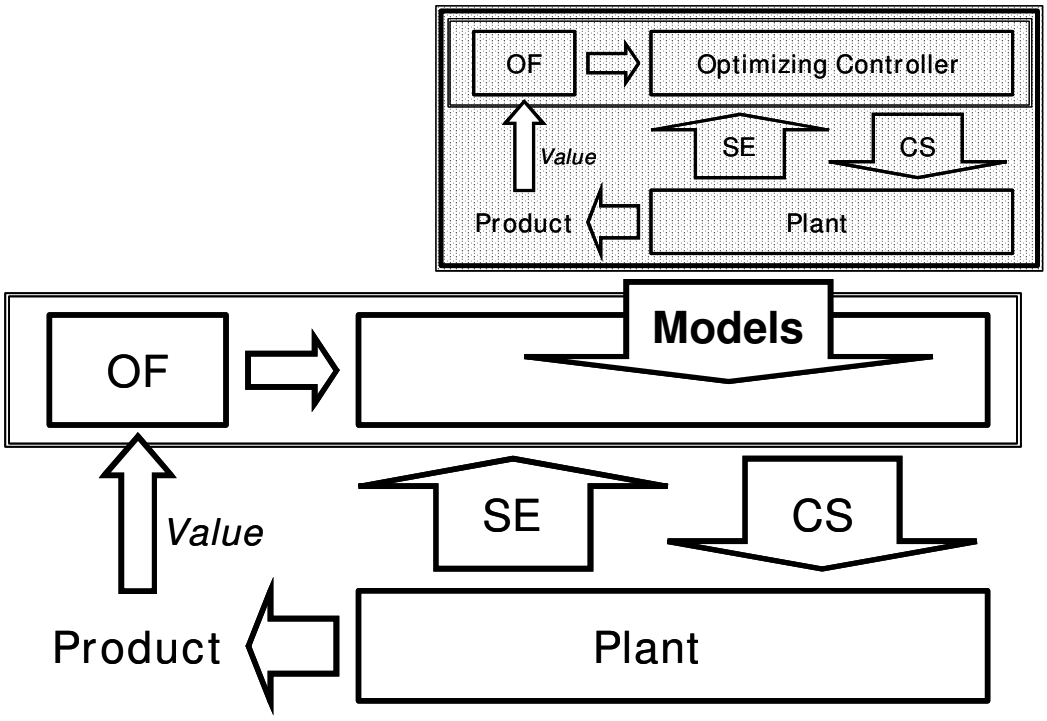

Figure A-3. The MPC Controller Contains Models of All Relevant System Entities, Including Itself

In practice, this event-driven-simulation at the heart of the predictive optimizer may take the form of Markov chains, Monte Carlo sampling experiments, or even closed-form solutions (when the models and drivers exhibit sufficient structure). Clearly, the models participating in this predictive process are the "crown jewels" of the MPC-based enterprise. Depending on the application, the availability and accuracy of state estimates to support these models may be a major performance driver. When this is the case, it may be possible to show that overall performance is enhanced by assigning constrained resources to **SE** activities (e.g., to maintain model currency and accuracy) – resources that would otherwise have been available for product delivery to customers.

A.2 Considerations

In addition to the conceptual satisfaction inherent in MPC, there are a number of other advantages deriving from a control theoretic approach. First is the ability to model the effect of feedback loops, and the potential occurrence of associated pathologies. From its inception, control theory has had to deal with feedback loops, and with the delays, uncertainties, reinforcements, and non-intuitive behaviors they engender. These issues are of most concern in a complex and highly dynamic environment that must react with agility (= low latency), and in which the end-to-end performance is dependent on, or conditioned by, a large number of intervening processes. It appears, based on our discussion in §1, that the remote sensing systems of the future (and the IAs that support them) will become increasingly dynamic, agile, and complex, so that a control theoretic solution may be indicated. Second, there are a number of system-level performance characteristics that are desirable, but that are difficult to model or optimize without notions from control theory. These include *stability* (and, in the case at hand, this means maintenance of expected behavior for standing customers in the face of unusual event-driven additional loading); *robustness* (that is, maintaining acceptable behavior in the face of uncertainties in state estimates and predictions); *chattering* (that is,

dampening excessive swings or oscillations when the system transitions across discrete threshold boundaries); and *observability* (that is, using models to infer and predict the state of elements that are not directly observable). And third, an optimizing controller of the sort described here ultimately traces all options and decisions up to the top-level objective function (tied, it is hoped, to value of delivered product). This means that it is possible, for example, to do sensitivity analysis, determining what factors in the space of decision control variables are most important *given the current circumstances*. The idea, here, is that the marginal utility (= value gradient) can vary greatly from component to component, depending on current circumstances. If we refer to this structural response of the **OF** to variations in control signal as the *value landscape* of the system at any point in time [34, 33], the ability to generate and analyze the value landscape can have significant operational utility. For example, under normal operating conditions, it may be that there is considerable flexibility in selecting a plan or schedule: many control options are virtually indistinguishable in terms of expected value. In such a case, Archive management may choose to impose additional factors or constraints, confident that top-level performance measures will not be penalized. In another scenario, the analysis may reveal that one or two factors are highly significant, and should be the principle focus of current attention. Knowing this, operational personnel might initiate additional modeling or knowledge extraction algorithms focused on these key factors, both the further understand the problem, and to develop appropriate responses (see, for example, the discussion under playbooks in §2 above).

Despite these advantages, a balanced assessment must consider the costs and risks associated with such an approach. In §3 of the text, we have discussed technical challenges. Here, we note three issues that a MPC approach must address. One is developmental cost and risk. If an archive can operate adequately using simple (or even legacy) algorithms, it becomes hard to justify the increased cost and technical risk associated with a more complex MPC approach. A second issue concerns computational complexity and latency. How long will the optimization routines take to execute, and what impact will that added latency have on end-to-end performance? Along similar lines, are they scalable and/or parallelizable, thereby enabling a latency-vs-hardware cost tradeoff? Third and finally, given the complexity of an enterprise such as an IA and the collection system in which it is embedded, it is very unlikely that closed-form solutions can be produced. Rather, heuristics will be required along with sub-optimal approximations to NP-complete problem formulations. In such cases, questions arise regarding how much modeling error is being introduced, and how much performance given up, in order to achieve a computationally reasonable instantiation. An "optimal in theory" problem formulation may give rise to a "sub-optimal in practice" solution due to cost and latency constraints. In this connection, it is useful to note that algorithms are available that can adjust to the available time: increased reliance on heuristics when time is short; but moving along the scale of increasing fidelity and optimization as time constraints are relaxed[39,39].

REFERENCES

[1] Ramapriyan, H. K., G. McConaughy, C. Lynnes, S. Kempler, K. McDonald, R. Harberts, L. Roelofs, and P. Baker, 2002. "Conceptual Study of Intelligent Archives of the Future", Report prepared for the Intelligent Data Understanding program, 39 p., http://daac/IDA/IA_report_8-27-02_baseline.pdf.

[2] Kubiatowicz, J. and D. Anderson, 2002. "The Worldwide Computer", ScientificAmerican.com, March 2002.

[3] Gibson, T. and E. Miller, 1999. "An Improved Long-Term File Usage Prediction Algorithm," Proceedings of the 24th Annual International Conference on Computer Measurement and Performance (CMG '99), Reno, NV, December 1999, p. 639–648.

[4] Chavarría-Miranda, D. G., J. M. Mellor-Crummey, and T. Sarang, 2001. "Data-Parallel Compiler Support for Multipartitioning", Euro-Par 2001, p. 241-253.

[5] Morse, H. S., "Stabile and agile scheduling of overhead ISR assets," *Proc. Symposium on Advances in Enterprise Control I*, 15–16 November 1999, San Diego, CA, sponsored by DARPA-ISO, p. 191-196.

[6] Allgower, F., T.A. Badgwell, J.S. Qin, J.B. Rawlings and S.J. Wright, 1999. "Nonlinear predictive control and moving horizon estimation – an introductory overview," *Advances in Control: Highlights of ECC '99*, Springer-Verlag, p. 391–449.

[7] Yoon, T. W. and D. W. Clark, 1995. "Observer design in receding-horizon predictive control," *International Journal of Control*, **61**(1), p. 171-191.

[8] Cristianini, N. and J. Shawe-Taylor, 2000. *An Introduction to Support Vector Machines*, Cambridge University Press.

[9] Fayyad, U. M., G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*, AAAI Press.

[10] Chen, Z., 2001. *Data Mining and Uncertain Reasoning: An Integrated Approach*, Wiley

[11] Almond, R. G., 1995. *Graphical Belief Modeling*, Chapman and Hall.

[12] Pearl, J., 1988. *Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann.

[13] Jang, J.-S. R., C.-T. Sun, and E. Mizutani, 1997. *Neuro-Fuzzy and Soft Computing*, Upper Saddle River, NJ: Prentice-Hall.

[14] Goodwin, G. and K. Sin, K., 1989. *Adaptive Filtering, Prediction, and Control*, Prentice-Hall.

[15] Mosca, E., 1995. *Optimal, Predictive, and Adaptive Control*, Prentice-Hall.

[16] Sastry, S. and M. Bodson, 1989. *Adaptive Control: Stability, Convergence and Robustness*, Prentice-Hall.

[17] Mayne, D. Q. J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, 2000. "Constrained model predictive control: stability and optimality," *Automatica* **36**, p. 789-814.

[18] Heise, S. A., 1994. "Stability of Constrained MBPC using and internal model control structure," in *Advances in Model Based Predictive Control*, D. Clark Ed., Oxford University Press.

[19] Lee, J., H. M. Morari and C. E. Garcia, 1994. "State-Space Interpretation of Model Predictive Control," *Automatica* **30** (4), p. 707-717.

[20] Casandros, C. and Wardi, Y., 1999. "Enterprise Engineering: A Framework for Optimal Control of Hybrid Dynamical Systems," *Proc. Symposium on Advances in Enterprise Control I*, 15 – 16 November 1999, San Diego, CA, sponsored by DARPA-ISO, p. 129-135.

[21] Kott, A. and Sircar, S., "Enterprise architecture analysis using an architecture description language," *Proc. Symposium on Advances in Enterprise Control II*, 10 – 11 July 2000, Minneapolis, MN, sponsored by DARPA-ISO, p. 195-202.

[22] Bertsekas, D., *Dynamic Programming and Optimal Control* (I and II), 1995, Athena Scientific.

[23] Bertsekas, D. and Shreve, S., *Stochastic Optimal Control: The Discrete-Time Case*, 1996, Athena Scientific.

[24] Bertsekas, D. and Castanon, D., "Dynamic programming methods for adaptive multi-platform scheduling in a risky environment," *Proc. Symposium on Advances in Enterprise Control II*, 10 – 11 July 2000, Minneapolis, MN, sponsored by DARPA-ISO, p. 121-128.

[25] Gomes, C., B. Selman, N. Crato, and H. Kautz, 2000. "Heavy-tailed phenomena in satisfiability and constraint satisfaction problems," *J. of Automated Reasoning*, **24**(1), p. 67-100

[26] Gomes, C. and B. Selman., 1999. "On the fine structure of large search spaces," *Proc. ICTAI'99*, Chicago, IL, November 1999.

[27] Gomes, C. and B. Selman, 1997. "Problem structure in the presence of perturbations," *Proc. of the Fourteenth National Conference on Artificial Intelligence* (AAAI-97), New Providence, RI., 1997

[28] Adams, M., O.M. Deutsch, W. D. Hall, R. H. Hildebrant, W. R. Kreamer, M. W. McConley, and H. F. Vuong, 2000. "Closed-loop operation of large-scale enterprises: application of a decomposition approach," *Proc. Symposium on Advances in Enterprise Control II,* 10 – 11 July 2000, Minneapolis, MN, sponsored by DARPA-ISO, p. 203-212.

[29] Adams, M. and W. Hall, 1999. "Closed-loop, hierarchical control of military air operations," *Proc. Symposium on Advances in Enterprise Control I*, 15 – 16 November 1999, San Diego, CA, sponsored by DARPA-ISO, p. 245-250.

[30] Kott, A. and B. Krogh, 1999. "Toward a catalog of pathological behaviors in complex enterprise control systems," *Proc. Symposium on Advances in Enterprise Control I*, 15 – 16 November 1999, San Diego, CA, sponsored by DARPA-ISO, p. 185-190.

[31] Heise, S. A. and H. S. Morse, 2000. "Agile Control of Military Operations", *Proc. 39th IEEE Conference on Decision and Control*, 12-15 December 2000, Sydney, Australia.

[32] McEneaney, W., 1999. "Robust game-theoretic methods in filtering and estimation," *Proc. Symposium on Advances in Enterprise Control I*, 15 – 16 November 1999, San Diego, CA, sponsored by DARPA-ISO, p. 3-9.

[33] Zou, G. and Y. P. Gupta, 2000. "Robust model predictive control for uncertain step response," *Optimal Control: Applications and Methods*, **21**(4), p. 161-184.

[34] Gelb, A., 1988. *Applied Optimal Estimation*, M. I. T. Press.

[35] Soeterboek, R., 1992. *Predictive Control: A Unified Approach*, Prentice-Hall International.

[36] Gopal, V., 1999. "Mathematical programming approaches for optimization and control of hybrid systems," *Proc. Symposium on Advances in Enterprise Control I*, 15 – 16 November 1999, San Diego, CA, sponsored by DARPA-ISO, p. 111-120.

[37] Rabelo, L., D. Godbole, J, Jelinek and V. Gopal, 2000. "Multimodel predictive control: from air operations to enterprise optimization," *Proc. Symposium on Advances in Enterprise Control II,* 10 – 11 July 2000, Minneapolis, MN, sponsored by DARPA-ISO, p. 149-157.

[38] Garvey, A. and V. Lesser, 1993. "Design-to-Time Real-Time Scheduling," *IEEE Trans on Systems, Man, and Cybernetics*, **23**(6), 1491 – 1502.

[39] Horvitz, E., 1987. "Reasoning about Beliefs and Actions under Computational Resource Constraints," *Proc. of 1987 Workshop on Uncertainty in Artificial Intelligence*, Seattle, WA.